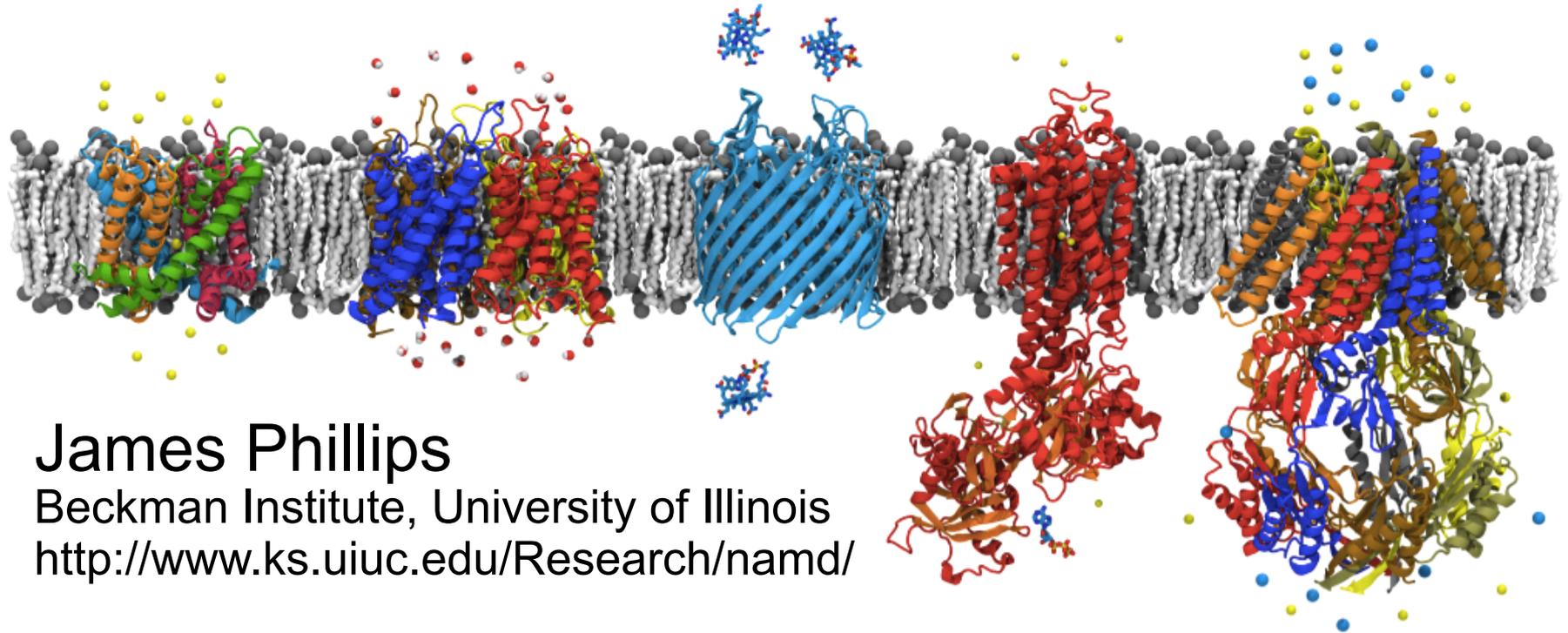


A Brief History of NAMD (and VMD)

Developing Sustainable Software in Academia



James Phillips

Beckman Institute, University of Illinois

<http://www.ks.uiuc.edu/Research/namd/>

NIH Biomedical Technology Research Center for Macromolecular Modeling and Bioinformatics

Developers of the widely used computational biology software VMD and NAMD

290,000 registered VMD users
72,000 registered NAMD users

600 publications (since 1972)
over 54,000 citations

5 faculty members
8 developers
1 systems
administrator
17 postdocs
46 graduate students
3 administrative staff

*Renewed 2012-2017
with 10.0 score (NIH)*

research projects include: virus
capsids, ribosome, photosynthesis,
protein folding, membrane reshaping,
animal magnetoreception

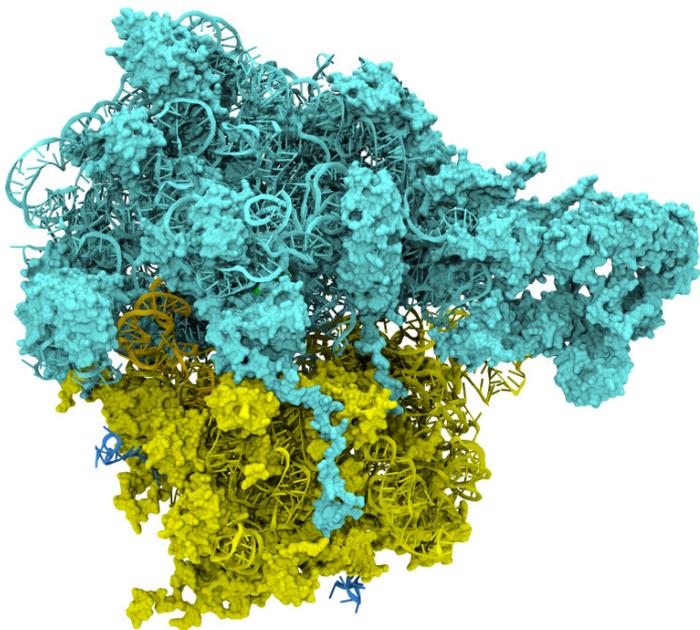
Achievements Built on People



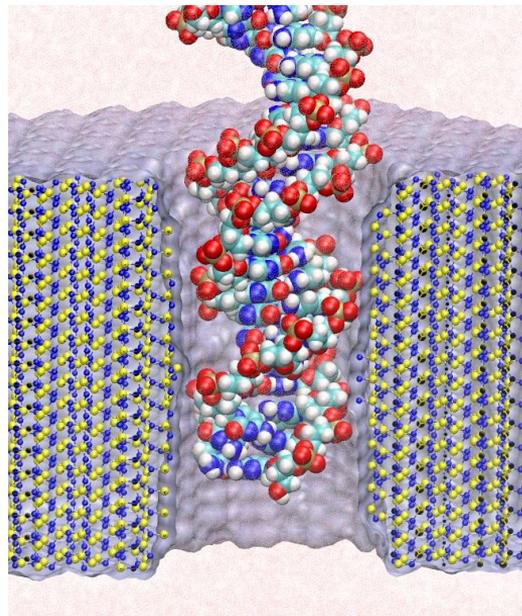
Tajkorshid, Luthey-Schulten, Stone, Schulten, Phillips, Kale, Mallon

Computational Microscopy

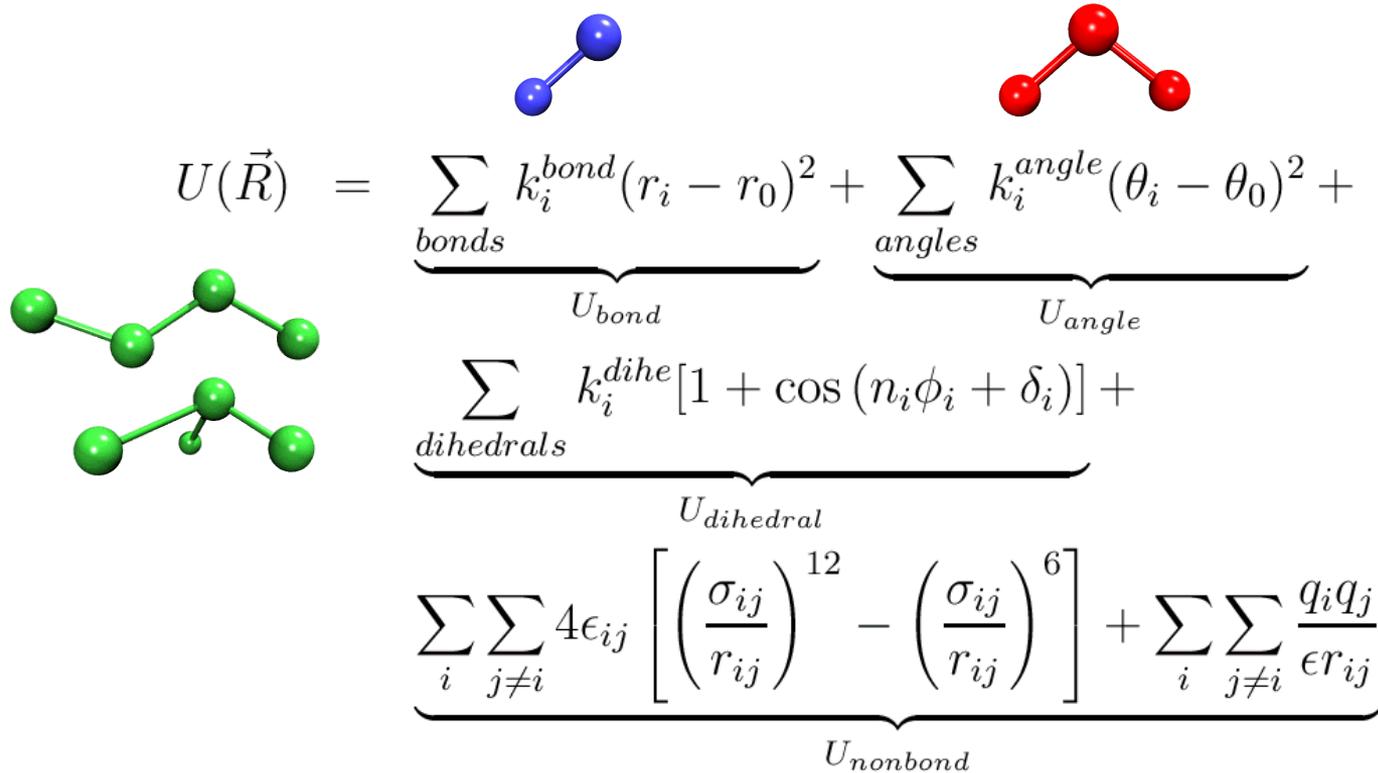
Ribosome: synthesizes proteins from genetic information, target for antibiotics



Silicon nanopore: bionanodevice for sequencing DNA efficiently



Molecular Mechanics Force Field



$$\begin{aligned}
 U(\vec{R}) = & \underbrace{\sum_{\text{bonds}} k_i^{\text{bond}} (r_i - r_0)^2}_{U_{\text{bond}}} + \underbrace{\sum_{\text{angles}} k_i^{\text{angle}} (\theta_i - \theta_0)^2}_{U_{\text{angle}}} + \\
 & \underbrace{\sum_{\text{dihedrals}} k_i^{\text{dihe}} [1 + \cos(n_i \phi_i + \delta_i)]}_{U_{\text{dihedral}}} + \\
 & \underbrace{\sum_i \sum_{j \neq i} 4\epsilon_{ij} \left[\left(\frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}} \right)^6 \right]}_{U_{\text{nonbond}}} + \sum_i \sum_{j \neq i} \frac{q_i q_j}{\epsilon r_{ij}}
 \end{aligned}$$

Classical Molecular Dynamics

Energy function: $U(\vec{r}_1, \vec{r}_2, \dots, \vec{r}_N) = U(\vec{R})$

used to determine the force on each atom: $m_i \frac{d^2 \vec{r}_i}{dt^2} = \vec{F}_i = -\vec{\nabla} U(\vec{R})$

Newton's equation represents a set of N second order differential equations which are solved numerically via the Verlet integrator at discrete time steps to determine the trajectory of each atom.

$$\vec{r}_i(t + \Delta t) = 2\vec{r}_i(t) - \vec{r}_i(t - \Delta t) + \frac{\Delta t^2}{m_i} \vec{F}_i(t)$$

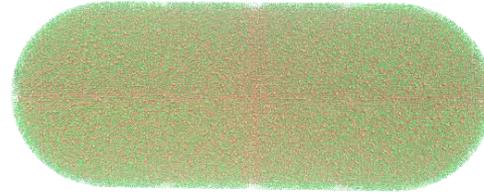
Small terms added to control temperature and pressure.

VMD – “Visual Molecular Dynamics”

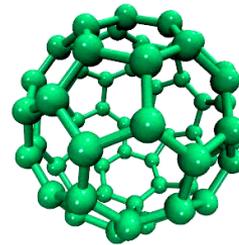
- Visualization and analysis of:
 - molecular dynamics simulations
 - particle systems and whole cells
 - cryoEM densities, volumetric data
 - quantum chemistry calculations
 - sequence information
- User extensible w/ scripting and plugins
- <http://www.ks.uiuc.edu/Research/vmd/>

Structural Similarity	
1hrc-a	GDVEKGGKKKIFVQKCAQCH
1ccr-a	ASFS...EAP...GNPKAGEKIFKTKCAQCH
1yea-a	AKEESTGFK...PSAKKGGATLFKTRCQQCH
5cyl-a	GDVAKGKKTFVQKCAQCH
1cyc-a	GDVAKGKKTFVQKCAQCH
1hrc-a	S...AP...GDPPVEGKHLFHTICITC
Sequence Similarity	
1hrc-a	GDVEKGGKKKIFVQKCAQCH
1ccr-a	ASFS...EAP...GNPKAGEKIFKTKCAQCH
1yea-a	AKEESTGFK...PSAKKGGATLFKTRCQQCH
5cyl-a	GDVAKGKKTFVQKCAQCH
1cyc-a	GDVAKGKKTFVQKCAQCH

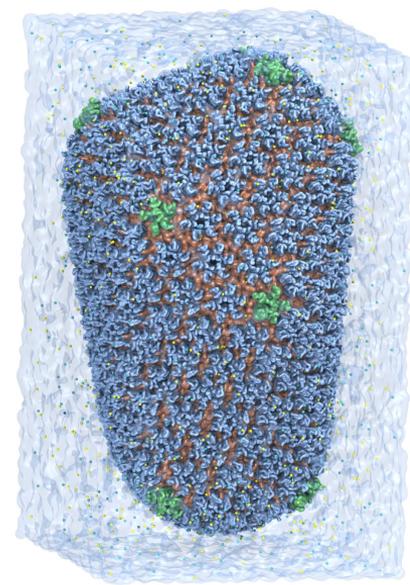
Sequence Data



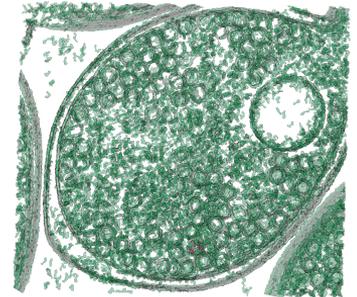
Whole Cell Simulation



Quantum Chemistry



MD Simulations



CryoEM, Cellular Tomography

NAMD Serves NIH Users and Goals

Practical Supercomputing for Biomedical Research

- 72,000 users can't all be computer experts.
 - 18% are NIH-funded; many in other countries.
 - 20,000 have downloaded more than one version.
 - 4000 citations of NAMD reference papers.
- One program available on all platforms.
 - Desktops and laptops – setup and testing
 - Linux clusters – affordable local workhorses
 - Supercomputers – free allocations on XSEDE
 - Blue Waters – sustained petaflop/s performance
 - GPUs - next-generation supercomputing
- User knowledge is preserved across platforms.
 - No change in input or output files.
 - Run any simulation on **any number of cores.**
- Available free of charge to all.



Hands-On Workshops



Oak Ridge TITAN

Long-term Charm++ Collaboration

- Illinois Parallel Programming Lab
 - Prof. Laxmikant Kale
 - charm.cs.illinois.edu
- Long standing collaboration
 - Since start of Center in 1992
 - Gordon Bell award at SC2002
 - Joint Fernbach award at SC12
- Synergistic research
 - NAMD requirements drive and validate CS work
 - Charm++ software provides unique capabilities
 - Enhances NAMD performance in many ways

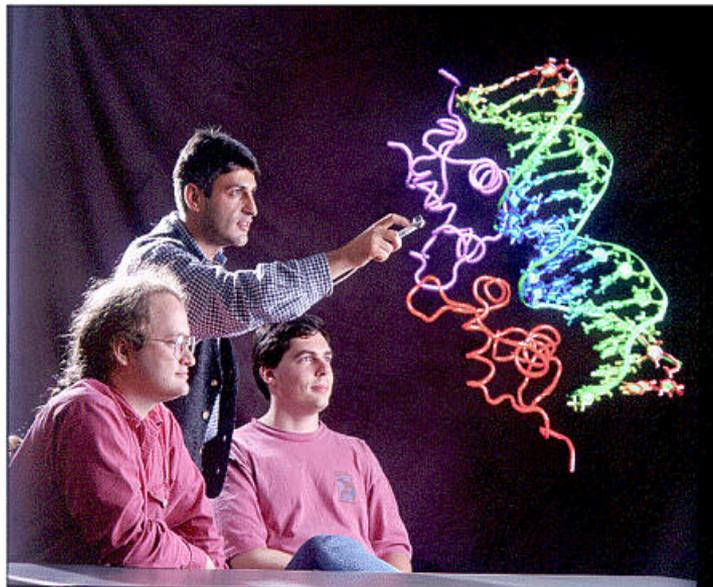
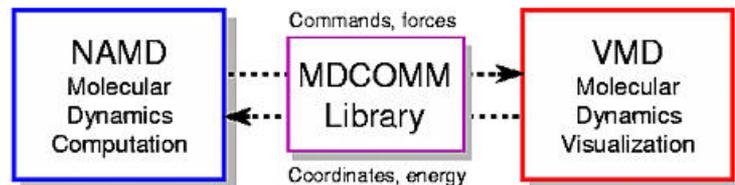


MDScope (1994)

- NAMD – dynamics on workstation clusters
 - PPL collaboration just started
 - Student-led revolt to write new code
 - Not A...’s Molecular Dynamics
- VMD – visualization
 - Originally VRChem for CAVE
 - NIH told us not to do visualization
 - Existing codes poor for trajectories
 - Scriptable interface, eventually Tcl
 - Andrew model: wrap other tools
 - Bill model: write code ourselves
- MDCOMM – steering communication
 - Collaboration with NCSA
 - Eventually replaced with raw sockets

MDScope

A Computational Environment for
Structural Biology



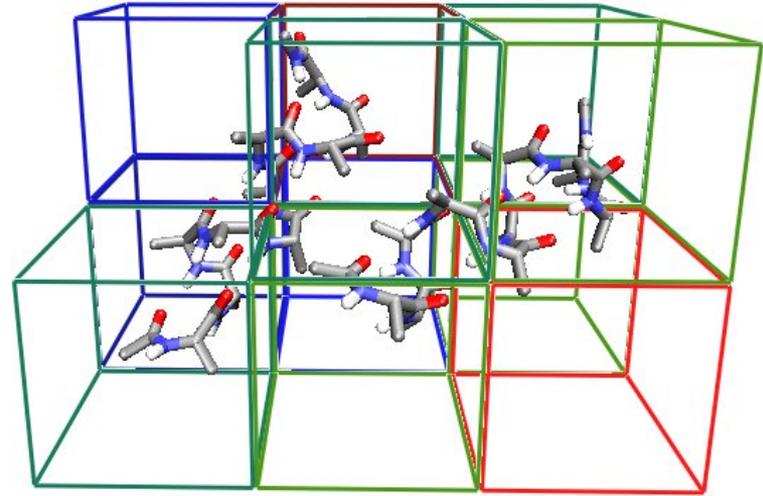
1994: Our First Cluster

- 14 HP workstations
- 125 MHz processor
- 128 MB memory
- 100 Mbit network
(optical ATM switch)
- ~ \$20K per processor
(cheaper than CM5!)



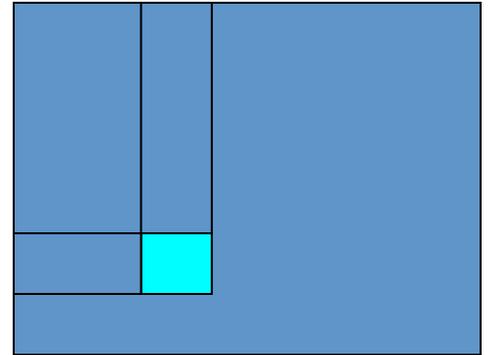
1994: Our First NAMD

- Written in C++
- Parallelized with PVM
- Spatial decomposition
- Message driven
- DPMTA electrostatics
- ~ 10,000 atom systems
- Up to ~ 8 processors

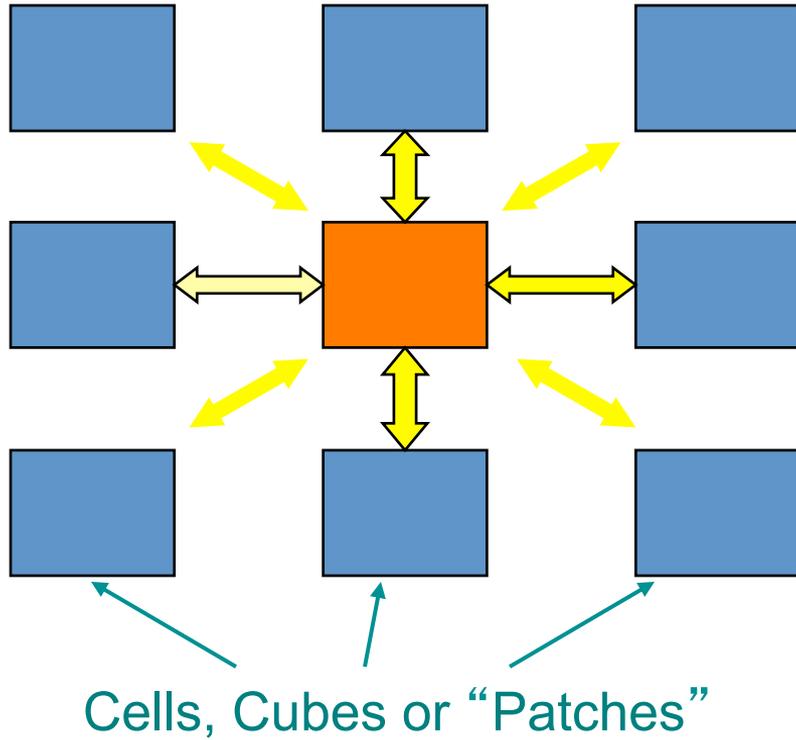


Poorly Scaling Approaches

- Replicated data
 - All atom coordinates stored on each processor
 - Communication/Computation ratio: $O(P \log P)$
- Partition the atom array across processors
 - Nearby atoms may not be on the same processor
 - C/C ratio: $O(P)$
- Distribute force matrix to processors
 - Matrix is sparse, non uniform
 - C/C Ratio: $O(\sqrt{P})$



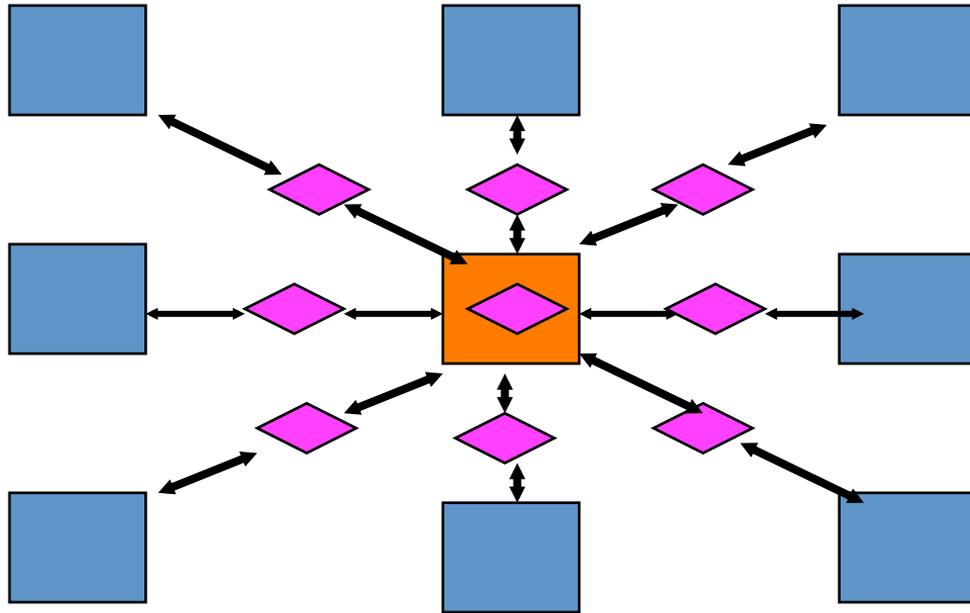
NAMD 1 Spatial Decomposition



- Atoms spatially distributed to cubes
 - Multiple cubes per processor.
 - Early example of virtualization!
- Size of each cube :
 - Just a larger than cut-off radius
 - Communicate only w/ neighbors
 - Work for each pair of neighbors
- C/C ratio: $O(1)$
- However:
 - Load Imbalance
 - Limited Parallelism

NAMD 2 Hybrid Decomposition

Kale *et al.*, *J. Comp. Phys.* 151:283-312, 1999.



- Spatially decompose data and communication.
- Separate but related work decomposition.
- “Compute objects” facilitate iterative, measurement-based load balancing system.

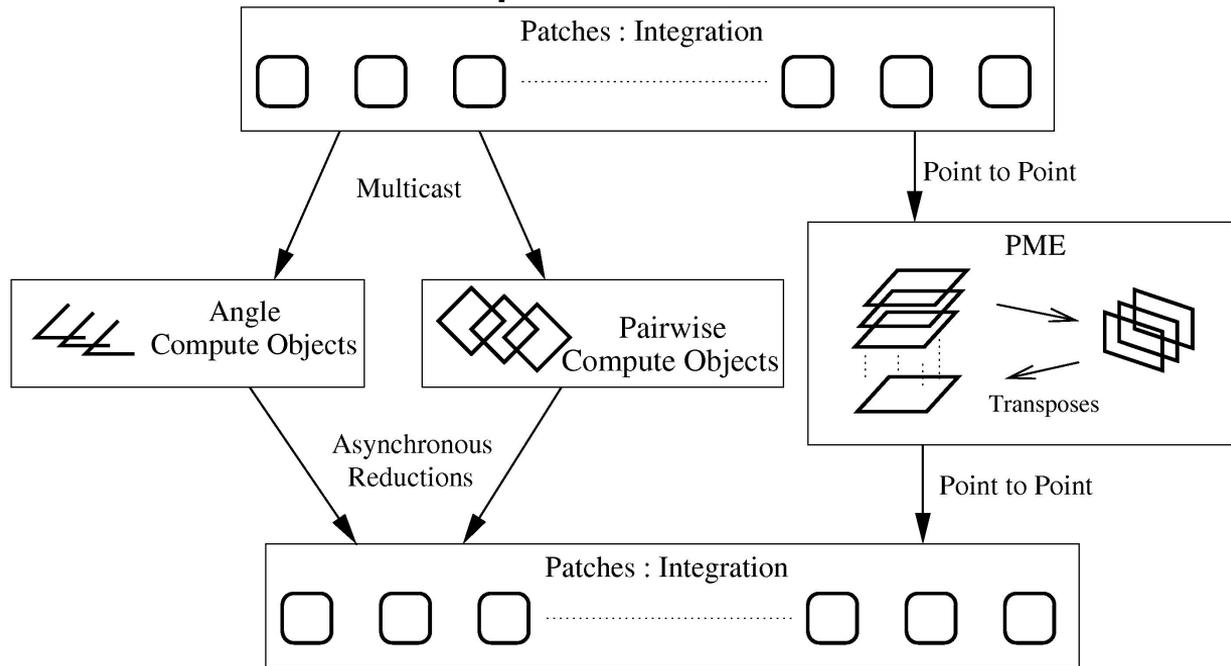
Implementation in 1997 Charm++

- Parallel C++ with *data driven* objects.
- Object groups:
 - Global object with a “representative” on each PE.
- Asynchronous method invocation.
- Prioritized scheduling of messages/execution.
- Measurement-based load balancing.
- Portable messaging layer.



NAMD Overlapping Execution

Phillips *et al.*, SC2002.



Objects are assigned to processors and queued as data arrives.

Coordinated Message Priorities

- Computes enqueued as messages
- Priorities based on critical path
 - Earlier step before later step
 - Earlier PME stages before later stages
 - Computes for remote patches before local
- Modules must be coordinated!
 - All priorities defined in `Priorities.h`
 - Diagnose and confirm using Projections

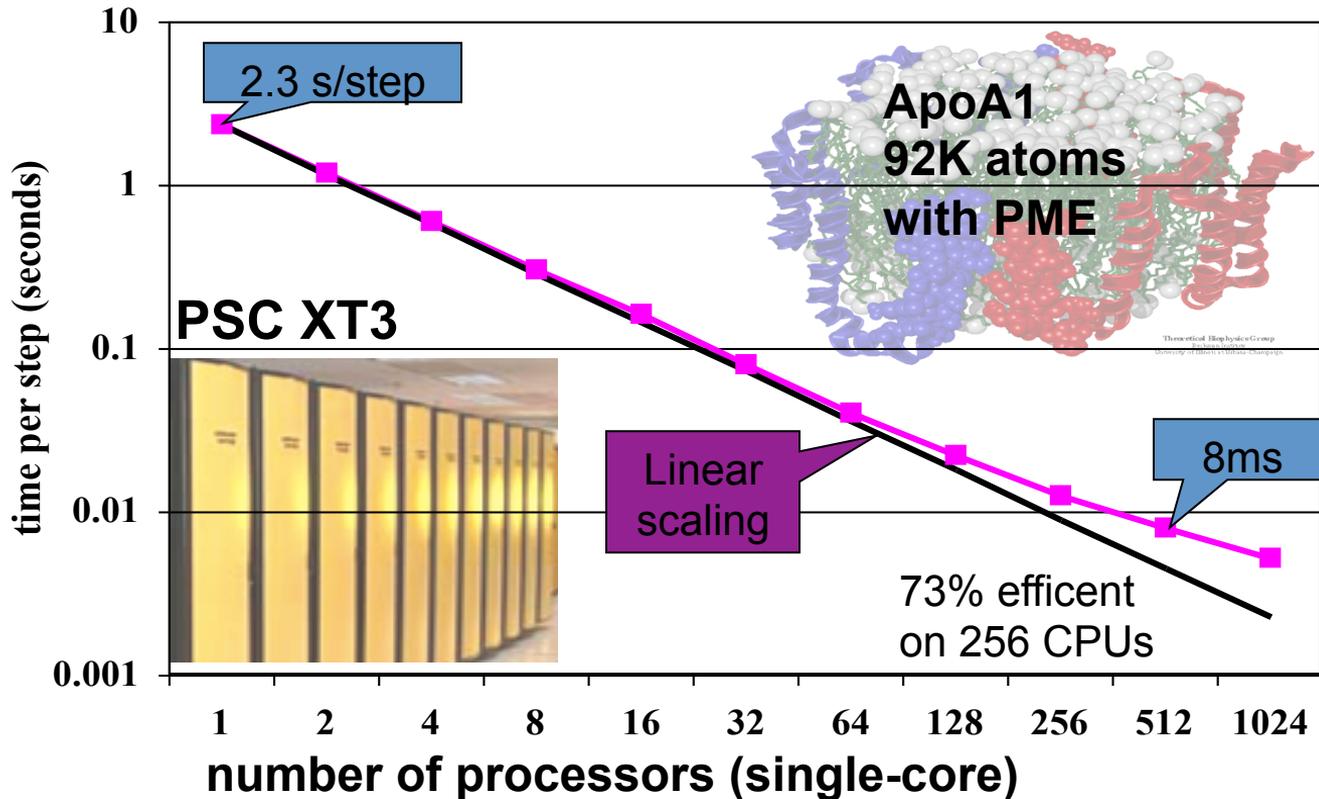


Startup Phases

- In parallel after reading input data on pe 0
- Used to load data, construct and link objects
- Phases separated by quiescence detections
- Simplest way to ensure side effects done
- Allows diagnosis of:
 - What operation is crashing or hanging
 - What operation is using memory or time

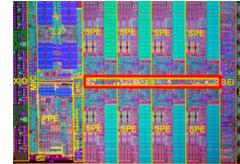


2006 NAMD Performance



Early Acceleration Options

- Outlook in 2005-2006:
 - FPGA reconfigurable computing (with NCSA)
 - Difficult to program, slow floating point, expensive
 - Cell processor (NCSA hardware)
 - Relatively easy to program, expensive
 - ClearSpeed (direct contact with company)
 - Limited memory and memory bandwidth, expensive
 - MDGRAPE
 - Inflexible and expensive
 - Graphics processor (GPU)
 - Program must be expressed as graphics operations



CUDA: Practical Performance

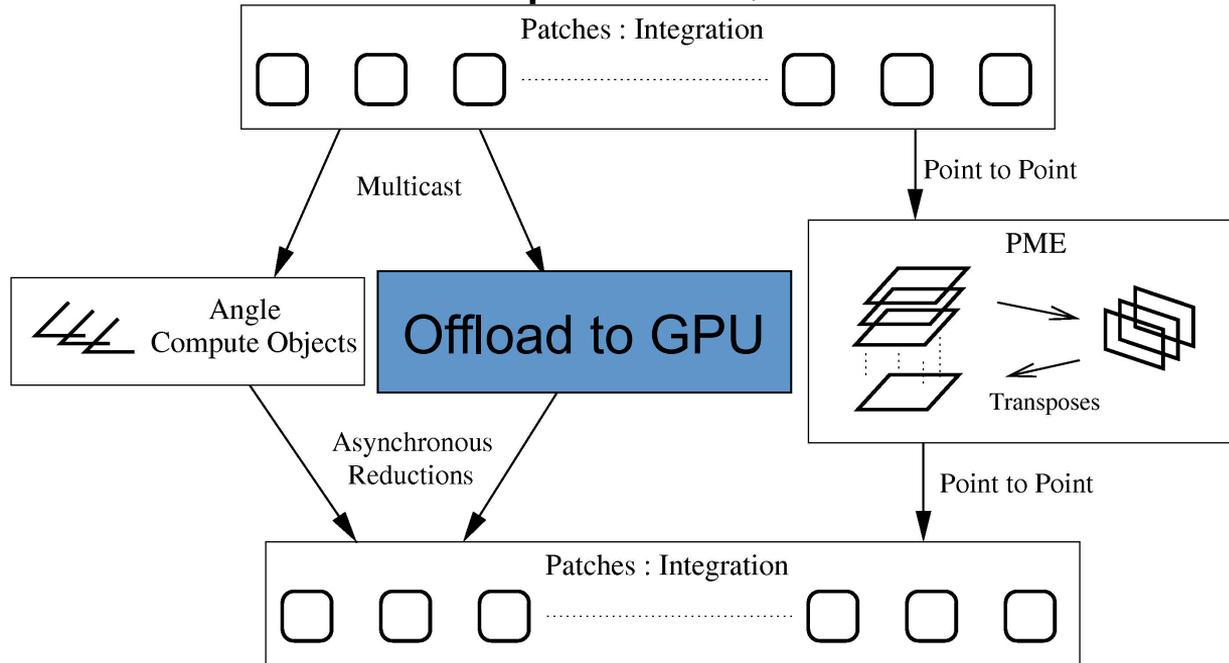
November 2006: NVIDIA announces CUDA for G80 GPU.

- CUDA makes GPU acceleration usable:
 - Developed and supported by NVIDIA.
 - No masquerading as graphics rendering.
 - New shared memory and synchronization.
 - No OpenGL or display device hassles.
 - Multiple processes per card (or vice versa).
- Center and collaborators make it useful:
 - Experience from VMD development
 - David Kirk (Chief Scientist, NVIDIA)
 - Wen-mei Hwu (ECE Professor, UIUC)



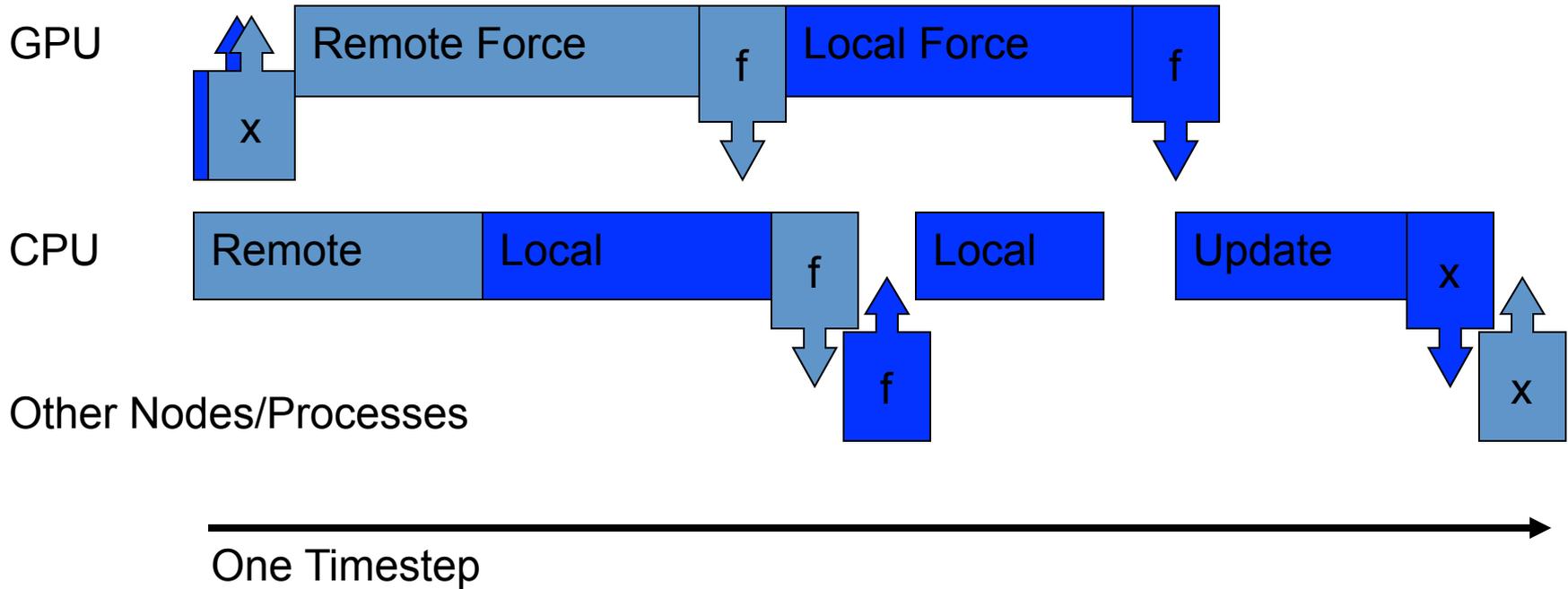
NAMD Overlapping Execution

Phillips *et al.*, SC2002.



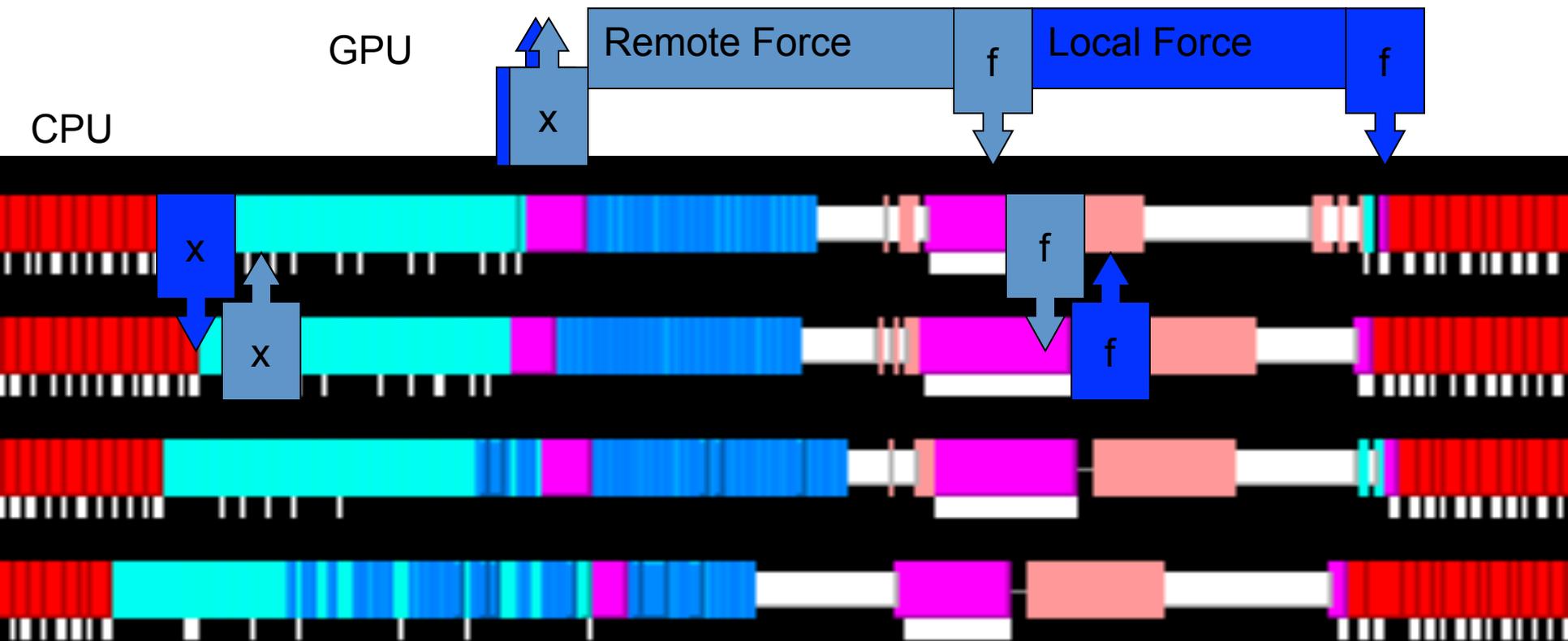
Objects are assigned to processors and queued as data arrives.

Overlapping GPU and CPU with Communication



Actual Timelines from NAMD

Generated using Charm++ tool "Projections" <http://charm.cs.uiuc.edu/>



Gearing Up for Petascale

- 2006 - NSF calls for 100 million atom simulation
 - Had just published million-atom virus simulation
- Issues to address:
 - Find scientific questions worthy of resources
 - Build model and initial coordinates
 - Store output trajectory
 - Analyze output trajectory
 - Scale NAMD to 100 million atoms
 - Scale NAMD to petascale machine(s)



NAMD for Large Systems

- Per-node memory usage
 - Exploit redundant structure
 - Pre-compressed static data
 - Distributed per-atom data
 - Special “memopt” build
 - Not all features supported
 - NAMD-only file formats
 - May change between versions
 - No VMD reader/writer - ever
- I/O performance
 - Data is relatively small
 - Parallelized POSIX I/O
 - Performance is just OK
 - New Charm++ I/O library being co-developed
- Parallelize load balancer
 - Local load balancing only

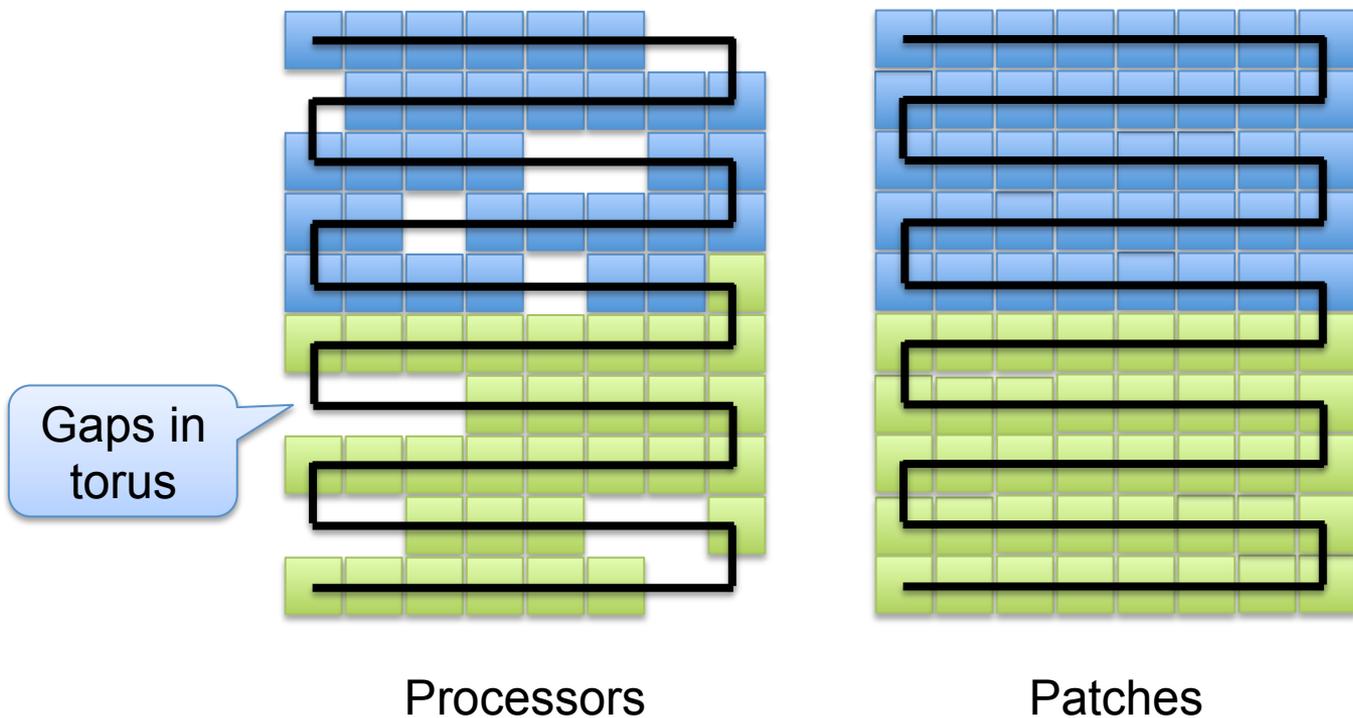


Irregular Torus Topologies

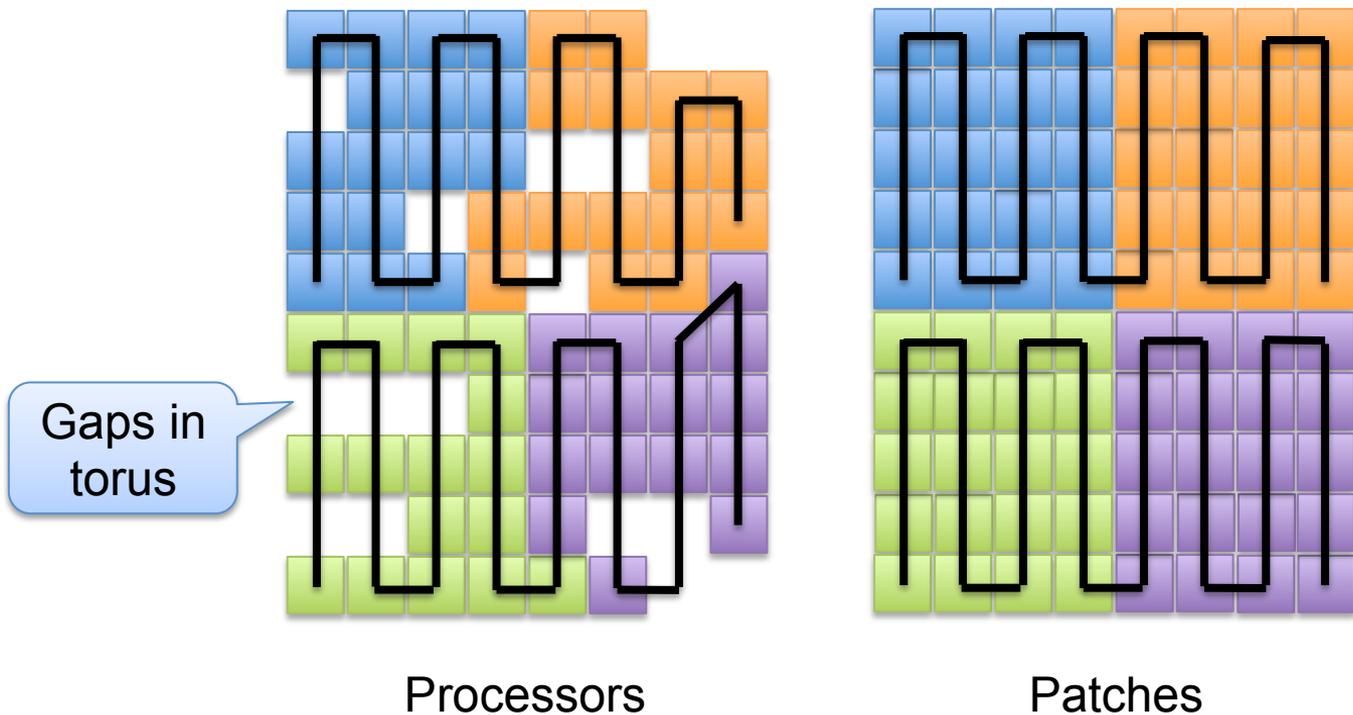
- IBM Blue Gene L/P/Q provide jobs with complete, regular, power-of-two torus.
- Cray XE/XK job topology is unpredictable.
 - Scheduler works around already running jobs.
 - May not be compact or contiguous.
 - New Blue Waters scheduler addresses this.
 - Even full-machine jobs skip over I/O nodes.



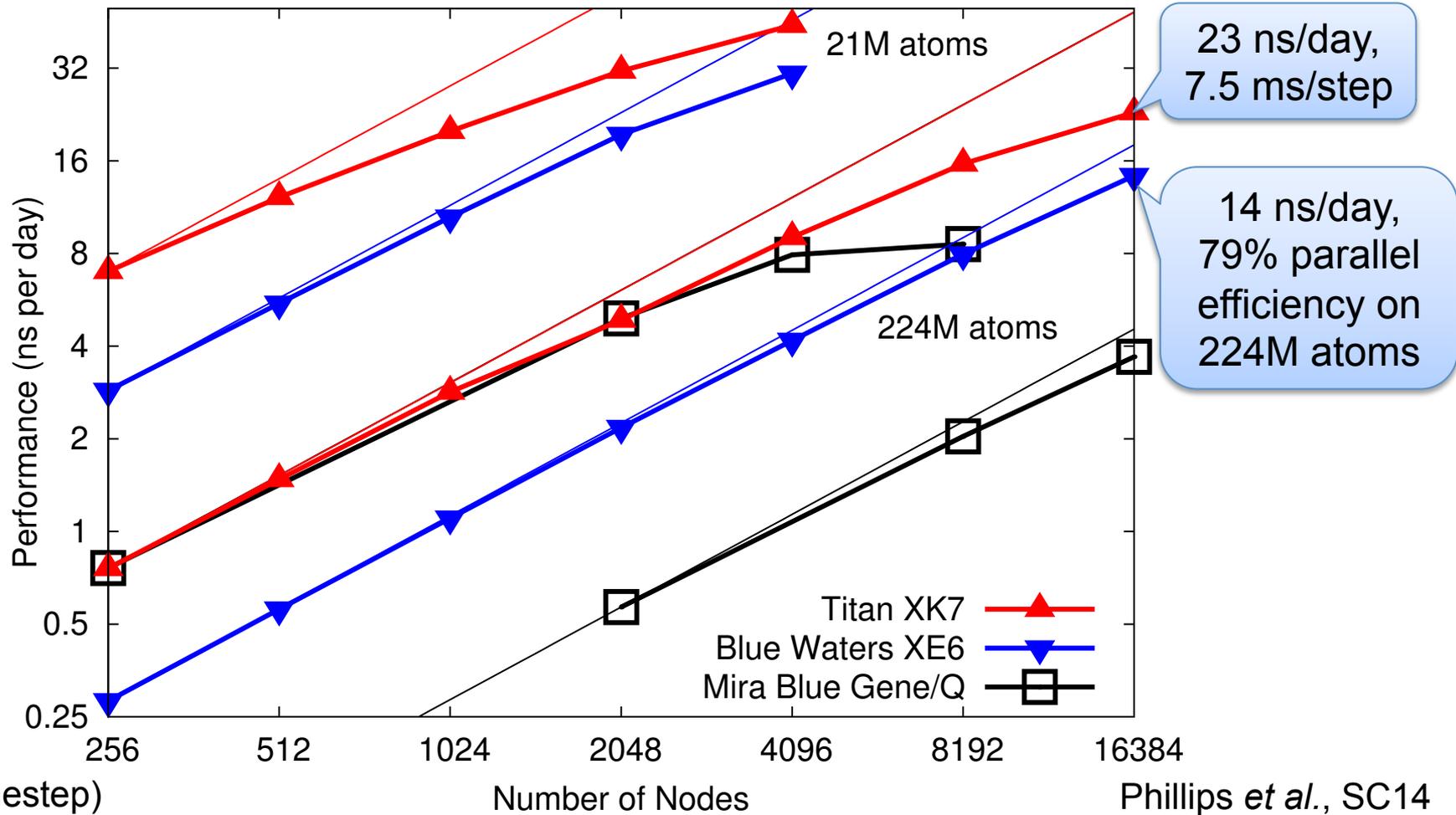
Mapping NAMD Spatial Domains



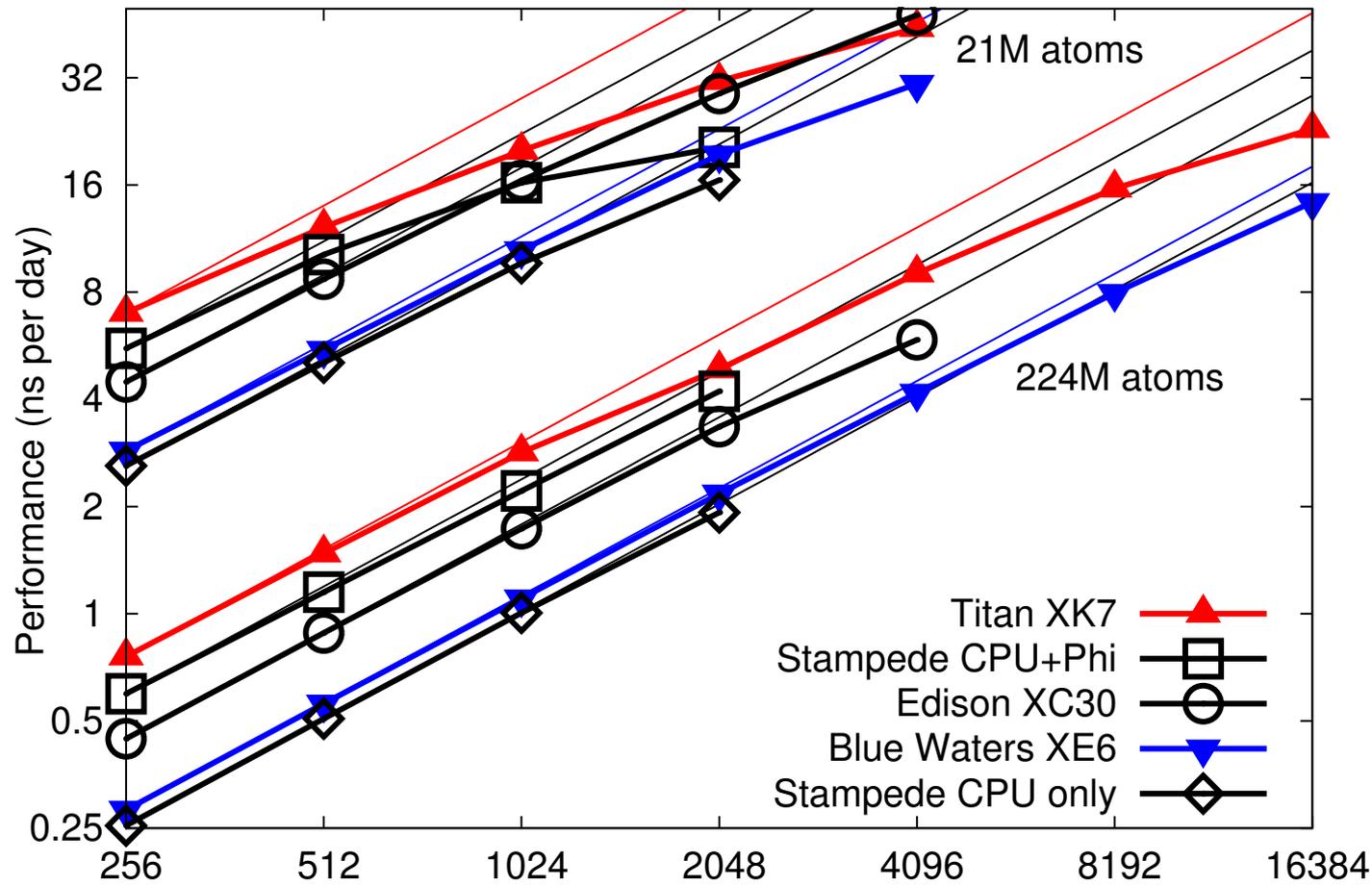
Mapping NAMD Spatial Domains



NAMD on Petascale Platforms



NAMD on Torus and Non-torus Networks

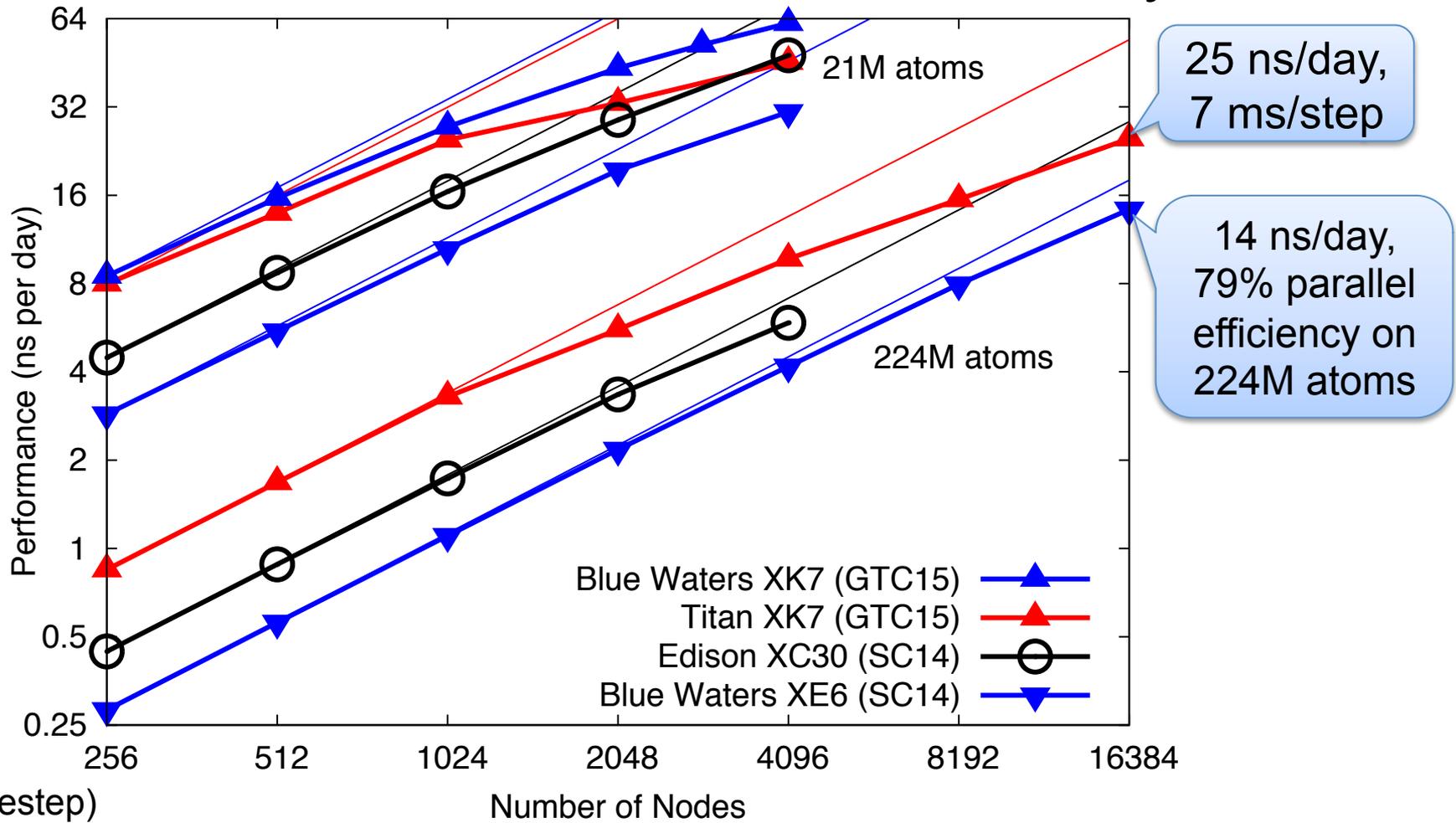


(2fs timestep)

Number of Nodes

Phillips *et al.*, SC14

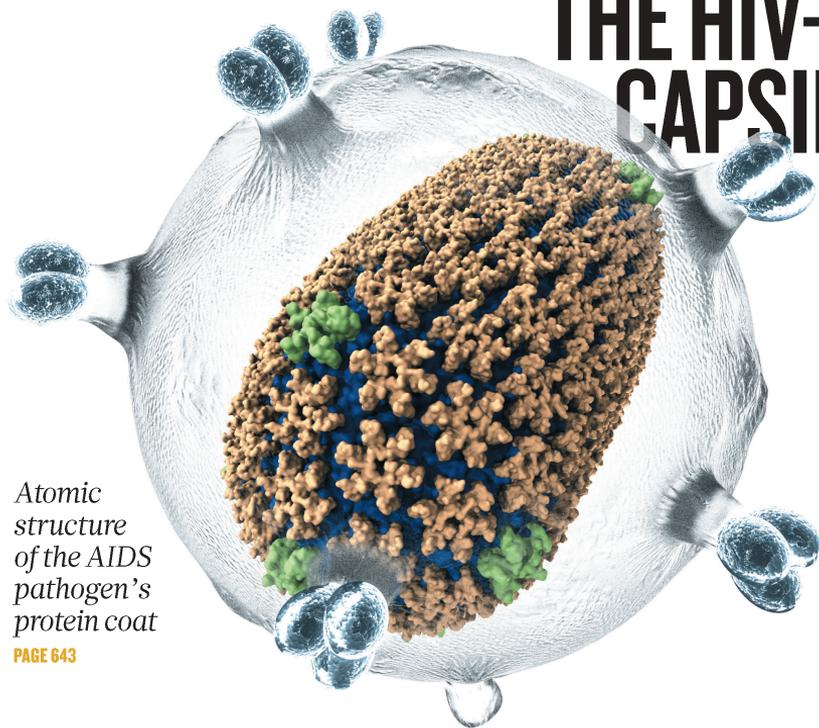
NAMD on Petascale Platforms Today



nature

THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE

THE HIV-1 CAPSID



Atomic
structure
of the AIDS
pathogen's
protein coat

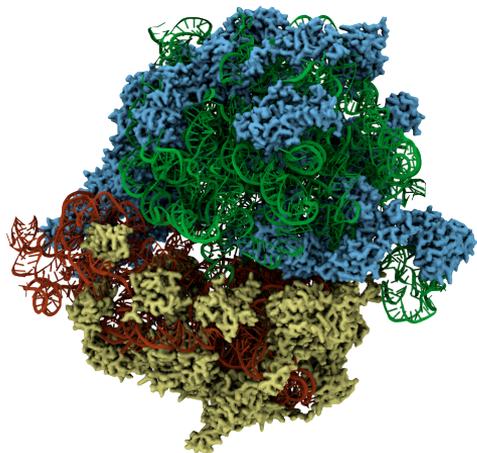
PAGE 643

2013 *HPCwire* Editors' Choice Award for Best Use of HPC in Life Sciences



Other Projects Using Petascale Computing

From cellular machines
to the pharmacy...



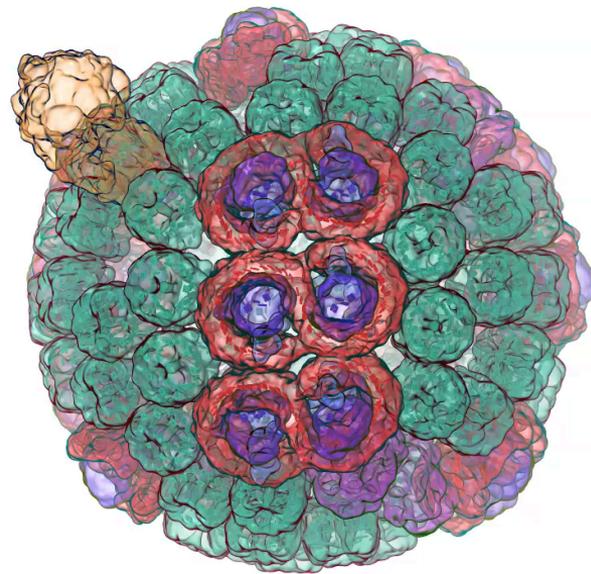
ribosome
3 M atoms, multiple copies

From woodchips to gasoline...



second-generation biofuels
> 10 M atoms

From solar energy to cellular fuel...



photosynthetic
chromatophore
100 M atoms

Looking Forward

- NERSC Cori / Argonne Theta (2016)
 - Knight's Landing (KNL) Xeon Phi
 - Single-socket nodes, Cray Aries network
 - Theta Early Science Project:
“Free Energy Landscapes of Membrane Transport Proteins”
- Oak Ridge Summit (2018)
 - IBM Power 9 CPUs + NVIDIA Volta GPUs
 - 3,400 fat nodes, dual-rail InfiniBand network
 - CAAR Project “Molecular Machinery of the Brain”
- Argonne Aurora (2018)
 - Knight's Hill (KNH) Xeon Phi

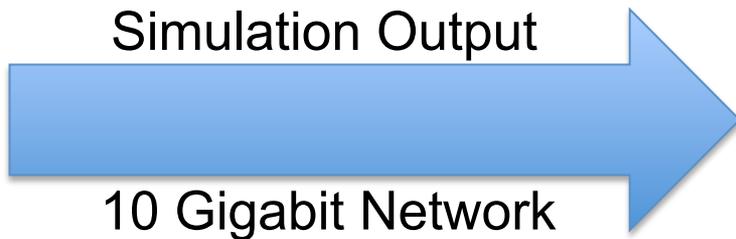


NIH Center Facilities Enable Petascale Biology

Over the past five years the Center has assembled all necessary hardware and infrastructure to prepare and analyze petascale molecular dynamics simulations, and ***makes these facilities available to visiting researchers.***



External Resources,
90% of our
Computer Power



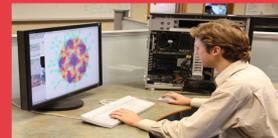
Petascale Gateway Facility

Storage



Compute

Visualization



High-End Workstations
Accessible to Visitors

Virtual Facilities Enable Petascale Anywhere



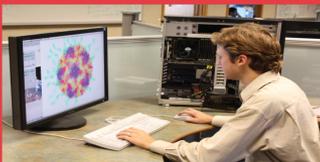
High-end visualization and analysis workstations currently available only in person at the Beckman Institute must be *virtualized and embedded at supercomputer centers.*

Storage



Compute

Visualization



Compressed Video



1 Gigabit Network

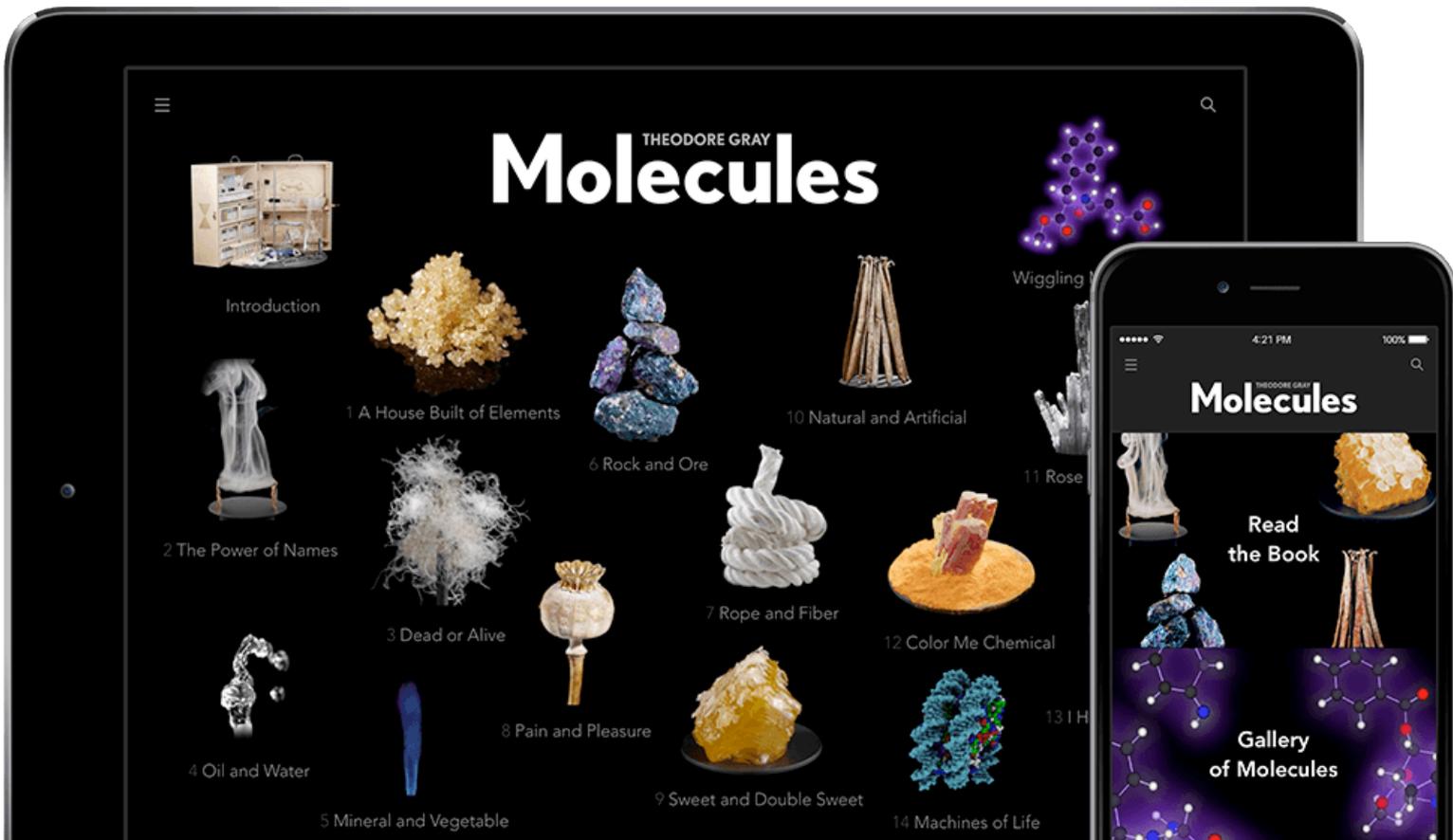


Distribution and Licensing

- Binaries and source code
 - Charm++ included
- Annual releases
- Nightly builds
- Registration required
- Public CVS access available
- Installed on supercomputers
- No redistribution
- Citation required
- Registration required
- Use for any purpose
- Combine up to 10% of source with at least 50% original code without restriction
- VMD plugins use BSD license



Special Licenses Are Possible



Support and Training

- Public mailing list
 - Other scientists know best
 - Archived and searchable
 - Social conventions apply
- Bug report emails
- Personal support
 - Driving projects
 - New capabilities
- Tutorials and Case Studies
 - Written by scientists
 - Focus on science problems
- Hands-on workshops
 - Taught by scientists
 - Several per year
 - Various locations
 - Requires only laptop



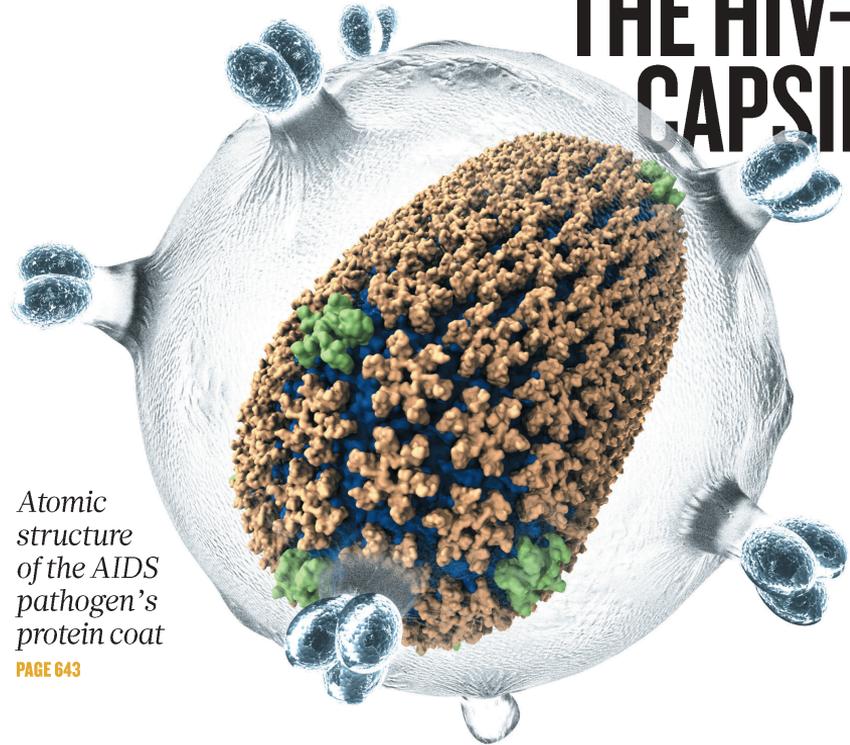
Advertising

- Demonstrate science that can be done with the code
- Make sure all contributors can share credit:
 - NCSA
 - NSF
 - DOE
 - NVIDIA
 - PPL
 - Collaborators

nature

THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE

THE HIV-1 CAPSID



*Atomic
structure
of the AIDS
pathogen's
protein coat*

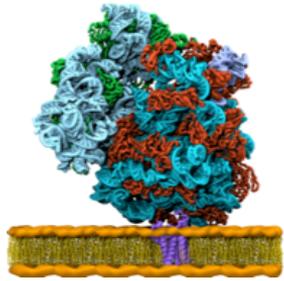
PAGE 643

Development Process/Philosophy

- Five-year funding cycle
 - Code, science, publish, proposal
- Evolutionary development
 - Fully functional code at all times
 - No stable/development branches
 - Large changes by refactoring only
- Simplify – don't manage
 - Separation of responsibilities
 - Alignment of incentives
 - Low coupling between people
- No code without an eager user
- No single-user features
- No schedules, no promises
- No design/code documentation
 - Source code must be **discoverable**
 - Use sandboxes to hide complexity
- Priorities and opportunities
 - Enabling new science
 - Supporting outside developers



Collaborative Driving Projects



1. Ribosome

R. Beckmann (U. Munich)
J. Frank (Columbia U.)
T. Ha (UIUC)
K. Fredrick (Ohio state U.)
R. Gonzalez (Columbia U.)

2. Blood Coagulation Factors

J. Morrissey (UIUC)
S. Sligar (UIUC)
C. Rienstra (UIUC)
G. Gilbert (Harvard)

3. Whole Cell Behavior

W. Baumeister (MPI Biochem.)
J. Xiao (Johns Hopkins U.)
C.N. Hunter (U. Sheffield)
N. Price (U. Washington)

4. Biosensors

R. Bashir (UIUC)
J. Gundlach (U. Washington)
G. Timp (U. Notre Dame)
M. Wanunu (Northeastern U.)
L. Liu (UIUC)

5. Viral Infection Process

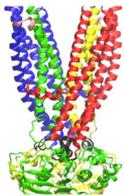
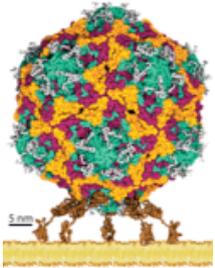
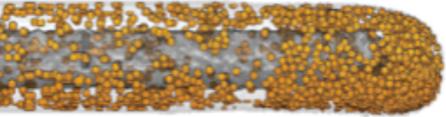
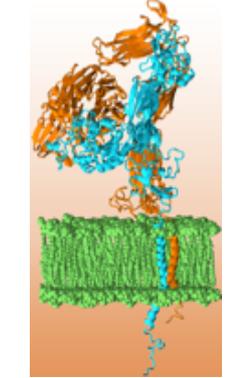
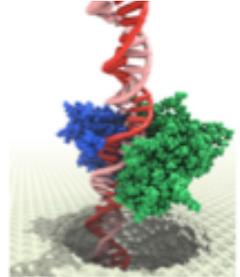
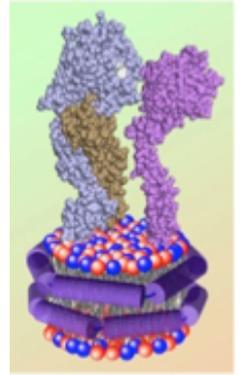
J. Hogle (Harvard U.)
P. Ortoleva (Indiana U.)
A. Gronenborn (U. Pittsburgh)

6. Integrin

T. Ha (UIUC)
T. Springer (Harvard U.)

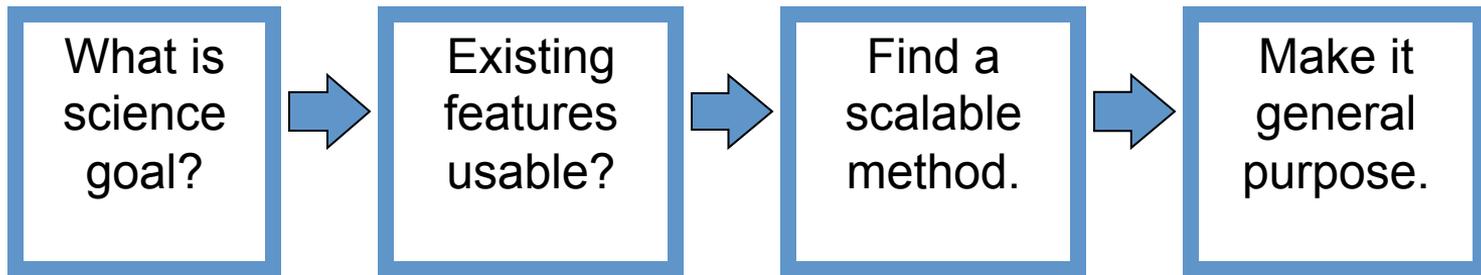
7. Membrane Transporters

H. Mchaourab (Vanderbilt U.)
R. Nakamoto (U. Virginia)
D.-N. Wang (New York U.)
H. Weinstein (Cornell U.)



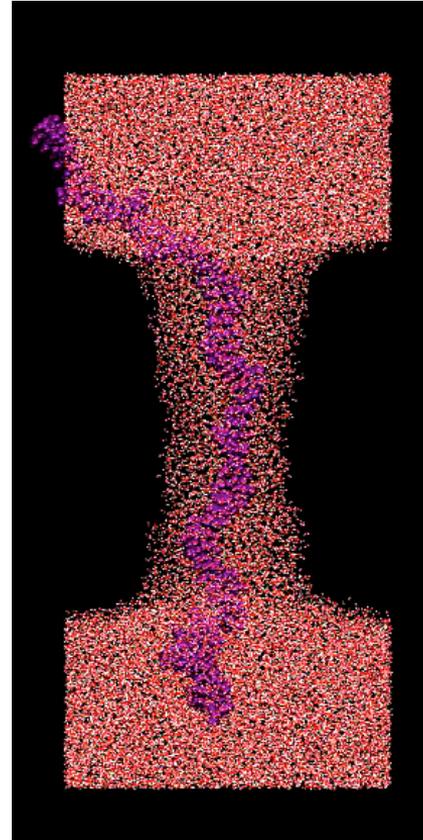
Collaborative Driving Projects

- Nearly every experimental collaboration relies on NAMD.
- High-end simulations push scaling efforts.
 - Try to anticipate needs: Million-atom virus just worked in 2006.
- Innovative simulations generate feature requests:



Adaptability Through Scripting

- Top-level protocols:
 - Minimize, heat, equilibrate
 - Simulated annealing
 - Replica exchange (originally via sockets)
- Long-range forces on selected atoms
 - Torques and other steering forces
 - Adaptive bias free energy perturbation
 - Coupling to external coarse-grain model
- Special boundary forces
 - Applies potentially to every atom
 - Several optimizations for efficiency
 - Shrinking phantom pore for DNA



Keys to NAMD Sustainability

- Stable funding and long-term staff
- Science applications drive features
- Large internal and external user base
- Enable user-driven customization
- Excellent software technology
- **Ensure all collaborators benefit**



Why NAMD and VMD Use Tcl

- History: Programs are ~20 years old.
- Maturity: Package management, portable.
- Stability: Interfaces haven't changed.
- Flexibility: Encapsulates mini-languages.
- Approachability: Looks like a simple scripting language, doesn't scare non-programmers.



Tcl Overview

- Variables: `$var $array($key) $array($i.field)`
- Strings: `abc 123 "$sub" {$nosub} [eval this]`
- Commands: `command $byvalue byname`
 - To create commands: `proc {args} {script}`
 - `upvar` and `uplevel` access calling namespace
 - Control structures are just commands
- Simple core enables great flexibility.



Latest Tcl/Tk Release: 8.6.4 (Mar 12, 2015)

- **Object Oriented Programming:** Gives Tcl a built-in object system that is fully dynamic, class-based, and includes advanced features such as meta-classes, filters, and mixins.
- **Stackless Evaluation:** Enables deep recursion in Tcl scripts. But there's more... This new implementation enables a collection of new commands, **coroutine**, **tailcall**, **yield**, and **yieldto** that provide profound new capabilities and models of concurrency to Tcl scripts.
- **Thread-enabled Operations:** A thread-enabled default build, a bundled **Thread** package, and new command `interp cancel` make Tcl 8.6 ready for your multi-threaded programming tasks.

Source: <http://www.tcl.tk/software/tcltk/8.6.html>



Examples of Tcl in VMD

- Atom selections – command as object:
 - set a [atomselect top “not water”]
 - \$a writepdb notwater.pdb
- Plugins written in Tcl, GUIs in Tcl/Tk.
 - Over 250,000 lines of Tcl code.
- Molecular structure building with psfgen:
 - segment SEG1 {first NTER; last CTER; pdb seg1.pdb}
- Save-state files are Tcl scripts.
 - Human-readable and modifiable.



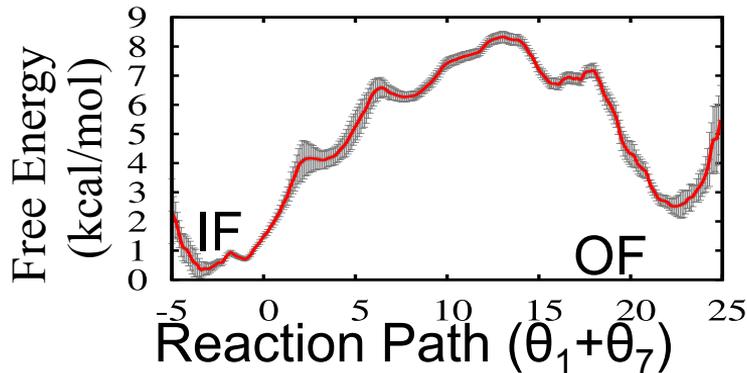
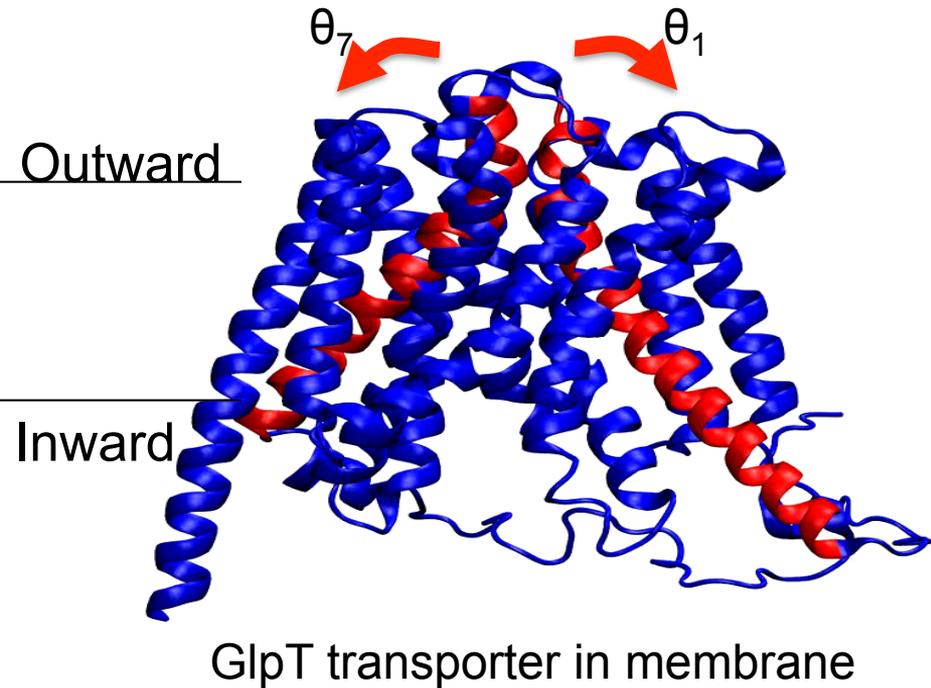
Tcl and Charm++ in NAMD

- Tcl runs on PE 0 only
 - Tcl parses config file until end or “run”
 - Send startup messages, run scheduler
 - Scheduler processes messages, starts run
 - At end of run, exit scheduler on quiescence
 - Tcl continues parsing config file...

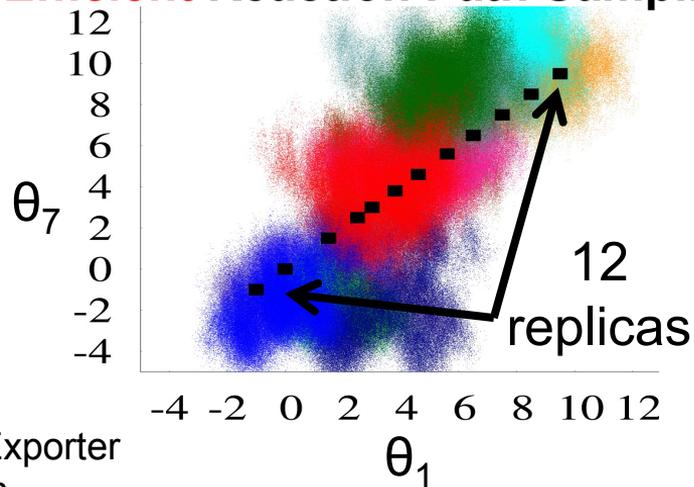


Replica Exchange Enables Advanced Sampling

Bias-Exchange Umbrella Sampling
on quaternion-based order parameters



Efficient Reaction Path Sampling



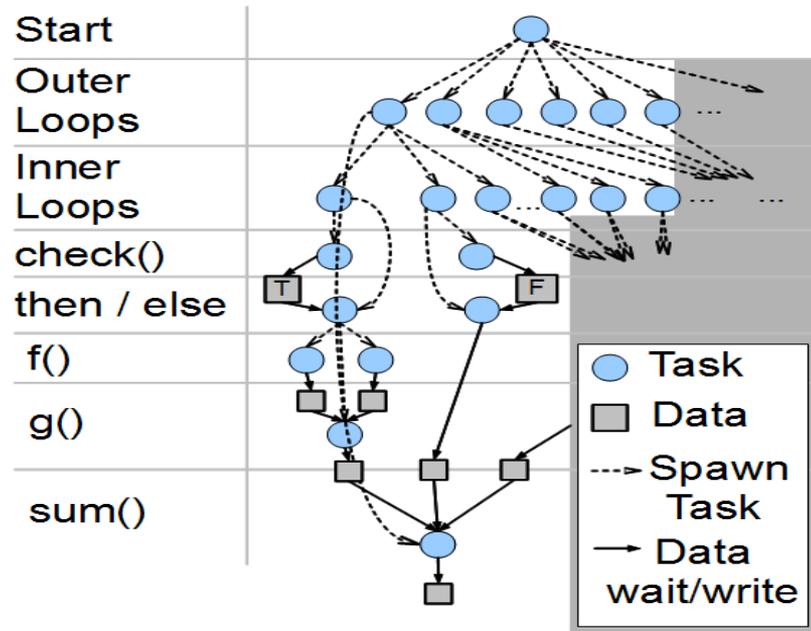
NAMD Replica Exchange Limitations

- One-to-one replicas to Charm++ partitions:
 - Available hardware must match science.
 - Batch job size must match science.
 - Replica count fixed at job startup.
 - No hiding of inter-replica communication latency.
 - No hiding of replica performance divergence.
- Can a different programming model help?



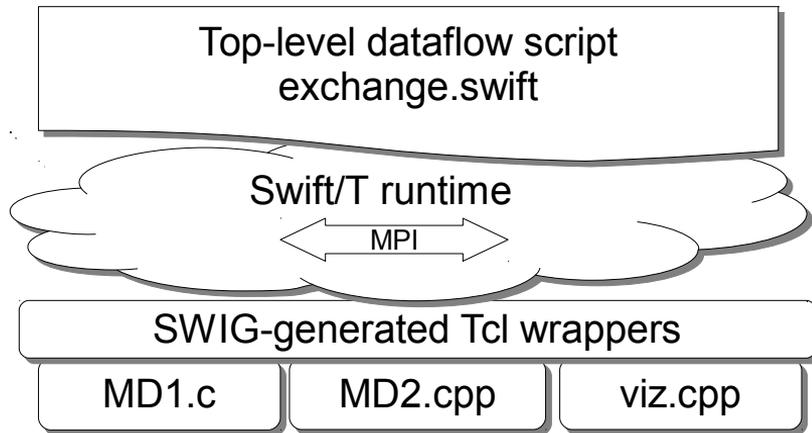
Swift/T: Fully parallel evaluation of complex scripts

```
int X = 100, Y = 100;
int A[][];
int B[];
foreach x in [0:X-1] {
  foreach y in [0:Y-1] {
    if (check(x, y)) {
      A[x][y] = g(f(x), f(y));
    } else {
      A[x][y] = 0;
    }
  }
  B[x] = sum(A[x]);
}
```

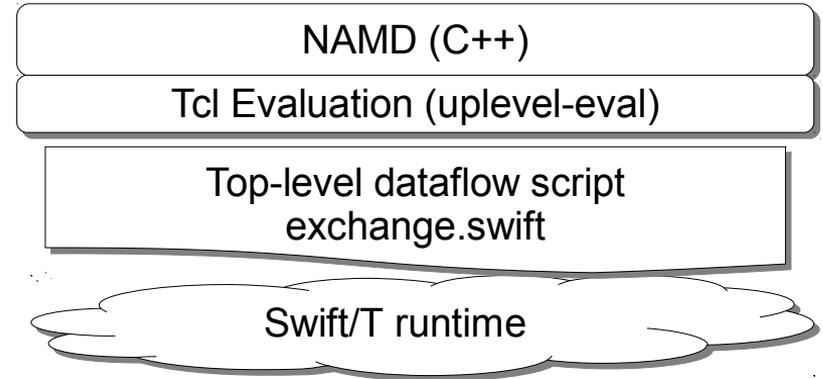


NAMD/VMD and Swift/T

Typical Swift/T Structure



NAMD/VMD Structure



Replica Exchange in Swift/T

```
foreach i in [0:num_replicas-1] {
  TEMPERATURE[i] = min_temp * exp(log(max_temp/min_temp)*(itof(i)/itof(num_replicas-1)));
  states[1][i], POTENTIAL[1][i] = run_t(ifile, i, 1, steps_per_run, TEMPERATURE[i], 300);
}
foreach f in [2:num_runs] {
  if ( f%%2 == 1 ) { sources[f][0] = 0; }
  if ( (num_replicas+f)%%2 == 1 ) { sources[f][num_replicas-1] = num_replicas-1; }
  foreach i in [f%%2+1:num_replicas-1:2] {
    BOLTZMAN = 0.001987191;
    dbeta = ((1.0/TEMPERATURE[i-1]) - (1.0/TEMPERATURE[i])) / BOLTZMAN;
    float delta = dbeta * (POTENTIAL[f-1][i] - POTENTIAL[f-1][i-1]);
    boolean doswap = (delta < 0.0) || (exp(-delta) > random());
    printf("frame %d reps %d %d swap %s\n", f, i-1, i, doswap);
    if ( doswap ) { sources[f][i] = i-1; sources[f][i-1] = i; } else { sources[f][i] = i; sources[f][i-1] = i-1; }
  }
  foreach i in [0:num_replicas-1] {
    int isrc = sources[f][i];
    states[f][i], POTENTIAL[f][i] = run_t(states[f-1][isrc], i, f, steps_per_run, TEMPERATURE[i], TEMPERATURE[isrc]);
  }
}
```



Replica Exchange in Tcl

```
while {$i_run < $num_runs} {
  run $steps_per_run
  save_array
  incr i_step $steps_per_run
  set TEMP $saved_array(TEMP)
  set POTENTIAL $saved_array(POTENTIAL)
  puts $history_file \
    "$i_step $replica(index) $NEWTEMP $TEMP $POTENTIAL"
  if {$i_run % 2 == 0} { set swap a; set other b }
  else { set swap b; set other a }
  set doswap 0
  if { $replica(index) < $replica(index.$swap) } {
    set temp $replica(temperature)
    set temp2 $replica(temperature.$swap)
    set BOLTZMAN 0.001987191
    set dbeta [expr ((1.0/$temp) - (1.0/$temp2)) / $BOLTZMAN]
    set pot $POTENTIAL
    set pot2 [replicaRecv $replica(loc.$swap)]
    set delta [expr $dbeta * ($pot2 - $pot)]
    set doswap [expr $delta < 0. || exp(-1.*$delta) > rand()]
    replicaSend $doswap $replica(loc.$swap)
    if { $doswap } { set rid $replica(index);
      set rid2 $replica(index.$swap) }
  }
}
```

```
if { $replica(index) > $replica(index.$swap) } {
  replicaSend $POTENTIAL $replica(loc.$swap)
  set doswap [replicaRecv $replica(loc.$swap)]
}
set newloc $r
if { $doswap } {
  set newloc $replica(loc.$swap)
  set replica(loc.$swap) $r
}
set replica(loc.$other) [replicaSendrecv \
  $newloc $replica(loc.$other) $replica(loc.$other)]
set oldidx $replica(index)
if { $doswap } {
  set OLDTEMP $replica(temperature)
  array set replica [replicaSendrecv [array get replica] $newloc $newloc]
  set NEWTEMP $replica(temperature)
  rescalelevels [expr sqrt(1.0*$NEWTEMP/$OLDTEMP)]
  langevinTemp $NEWTEMP
}
incr i_run
}
```

Portability



Aspects of Portability

- Operating systems
 - Linux, Mac, Windows
 - Lustre filesystem errors
 - System library and OS bugs
- CPU architecture
 - Compiler directives (e.g., ivdep)
 - OpenMP 4.0 “#pragma omp simd”
 - Occasional vector intrinsics
- Networks
 - Infiniband, Gemini, BG/Q
 - Offload to MPI/Charm++
 - Do not use Charm++ on MPI
 - Charm++ relatively fast to port
- Coprocessors
 - CUDA is mature and best in class
 - OpenCL isn't performance-portable
 - OpenACC doesn't support MIC
 - Intel offload directives only for MIC



Charm++ and MIC Options

- Naïve Offload
- Aggregated Offload
- Coprocessor Only
- Heterogeneous Cluster
- OpenMP Teams



Charm++ and MIC Options

- Naïve Offload
 - Just add offload directives
 - Close your eyes and trust the runtime
 - How many host threads can offload at once?
 - How do they share the MIC cores?
 - NAMD work grainsize is too small for MIC



Charm++ and MIC Options

- Aggregated Offload
 - Keep NAMD work decomposition
 - Collect and bulk-copy data
 - Bulk-launch tasks in single offload
 - Method initially developed for CUDA
 - NAMD MIC offload is clone of CUDA offload



Charm++ and MIC Options

- Coprocessor Only
 - Ignore host, run Charm++ scheduler on MIC
 - Requires all code to run acceptably on MIC
 - Not the case for KNC, hopefully for KNL
 - Fine for single card, maybe not for multi-node
 - Useful for preparing for KNL processor



Charm++ and MIC Options

- Heterogeneous Cluster
 - Treat host and MIC as two Charm++ nodes
 - Requires adapting to different core counts
 - Requires adapting to different core speeds
 - Performance ratio varies by function and **data**
 - E.g., self, face, and corner computes in NAMD
 - Full employment for computer scientists!



Charm++ and MIC Options

- OpenMP Thread Teams
 - Grainsize too large for single MIC thread
 - Grainsize too small for entire MIC
 - Let Charm++ control OpenMP thread teams
 - E.g, MIC = 15 Charm++ PEs
 - Each PE = 4 cores and 16 threads
 - Parallelize loops using OpenMP directives



MIC Vectorization Options

- Intrinsic – what we have
 - Written by David Kunzman of Intel
 - Currently in production
- Compiler – what we want
 - `#pragma [omp] simd assert`
 - Currently ~20% slower with refactored kernel
 - “One missed optimization” but improving fast
- ISPC – our backup plan
 - Similar to CUDA, consider if compiler fails



Conclusions and Ramblings

- Science, software, and supercomputing are all hard.
 - If you get good science from any supercomputer, you are winning.
- Solve problems you have before problems you might have.
 - Performance and correctness on one platform, then portability.
 - Complexity is forever - try the simplest thing that might work.
- Do look ahead - don't paint yourself into a corner.
 - But don't worry about things you don't yet understand well.
 - If you do get stuck, don't be afraid to refactor.
- If a problem has many solutions, it is probably unsolved.
 - But even a limited tool may work for your case.



Thanks to NIH, NSF, DOE, and 20 years of
NAMD and Charm++ developers and users.

